**ACTIVIDAD 15**

**Enunciado**

Realizar en Java un programa que cree 3 hilos y que su clase principal se llame Actividad15. Cada uno de ellos mostrará 5 mensajes por pantalla indicando su nombre y la hora actual. Entre mensaje y mensaje de un mismo hilo se esperará 1 segundo.

El resultado esperado sería equivalente al siguiente, teniendo en cuenta que el orden de aparición de los 5 mensajes de cada hilo puede variar de una ejecución a otra del programa y que la hora actual de ejecución se corresponderá con la hora en la que se ejecute dicho programa:

Hilo 1 - 18:54:03

Hilo 3 - 18:54:03

Hilo 2 - 18:54:03

Hilo 1 - 18:54:04

Hilo 3 - 18:54:04

Hilo 2 - 18:54:04

Hilo 1 - 18:54:05

Hilo 3 - 18:54:05

Hilo 2 - 18:54:05

Hilo 3 - 18:54:06

Hilo 1 - 18:54:06

Hilo 2 - 18:54:06

Hilo 3 - 18:54:07

Hilo 1 - 18:54:07

Hilo 2 - 18:54:07

**SOLUCION**

**//Actividad15.java**

public class Actividad15 {

public static void main(String[] args) {

//Creacion de Hilos

Hilo hilo1 = new Hilo();

Hilo hilo2 = new Hilo();

Hilo hilo3 = new Hilo();

//Nombres de los hilos

hilo1.setName("Hilo 1");

hilo2.setName("Hilo 2");

hilo3.setName("Hilo 3");

//Ejecucion de los hilos

hilo1.start();

hilo2.start();

hilo3.start();

//fin del programa

System.out.println("Fin del programa");

}

}

**//Hilo.java**

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

public class Hilo extends Thread{

private DateFormat df;

private Date date;

//Constructor para asignar el formato de la hora

public Hilo () {

df = new SimpleDateFormat("HH:mm:ss");

}

public void run() {

for (int i=1; i<=5;i++) {

//creamos una nueva fecha

date = new Date();

//imprimimos por pantalla el nombre y la hora

System.out.println(this.getName()+ " - "+ df.format(date));

try {

//Esperamos 1 segundo

sleep(1000);

} catch (InterruptedException e) {

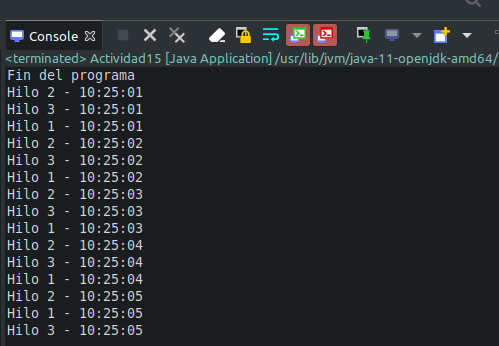
e.printStackTrace();

}

}

}

}



**ACTIVIDAD 16**

**Enunciado**

Modificar el siguiente programa en Java para conseguir el siguiente resultado:

HiloA contador vale 400

HiloB contador vale 100

La clase principal de este programa se llamará Actividad16 y la modificación a realizar requerirá sincronizar el objeto contador, de tal forma que el HiloA incremente 300 veces el contador, bloqueando el objeto, para que posteriormente el HiloB lo decremente.

El detalle del programa a modificar es el siguiente:

**public** **class** Actividad16 {

**public** **static** **void** main (String[] args) {

Contador cont = **new** Contador (100);

HiloA a = **new** HiloA("HiloA", cont);

HiloB b = **new** HiloB("HiloB",cont);

a.start();

b.start();

}

}

**class** Contador {

**private** **int** c=0;

Contador (**int** c) {

**this**.c = c;

}

**public** **void** incrementa() {

c=c+1;

}

**public** **void** decrementa() {

c=c-1;

}

**public** **int** getValor() {

**return** c;

}

}

**class** HiloA **extends** Thread {

**private** Contador contador;

**public** HiloA (String n, Contador c) {

setName(n);

contador=c;

}

**public** **void** run () {

**for** (**int** j=0; j < 300; j++) {

contador.incrementa();

**try** {

Thread.*sleep*(100);

} **catch** (InterruptedException e) {}

}

System.***out***.println(getName() + " contador vale" + contador.getValor());

}

}

**class** HiloB **extends** Thread {

**private** Contador contador;

**public** HiloB (String n, Contador c) {

setName(n);

contador=c;

}

**public** **void** run () {

**for** (**int** j=0; j < 300; j++) {

contador.decrementa();

**try** {

Thread.*sleep*(100);

} **catch** (InterruptedException e) {}

}

System.***out***.println(getName() + " contador vale" + contador.getValor());

}

}

**SOLUCION**

**//Actividad16.java**

package actividad16;

public class Actividad16 {

public static void main (String[] args) {

Contador cont = new Contador (100);

//Creamos los hilos con el contador inicialmente a 100

HiloA a = new HiloA("HiloA", cont);

HiloB b = new HiloB("HiloB",cont);

a.start();

b.start();

}

}

//Clase contador que incrementa o decrementa

class Contador {

private int c=0;

Contador (int c) {

this.c = c;

}

public void incrementa() {

c=c+1;

}

public void decrementa() {

c=c-1;

}

public int getValor() {

return c;

}

}

class HiloA extends Thread {

private Contador contador;

public HiloA (String n, Contador c) {

setName(n);

contador=c;

}

public void run () {

//Bucle para incrementar el contador en 300, sincronizado con el otro hilo

synchronized (contador) {

for (int j=0; j < 300; j++) {

contador.incrementa();

try {

Thread.sleep(100);

} catch (InterruptedException e) {}

}

System.out.println(getName() + " contador vale " + contador.getValor());

}

}

}

class HiloB extends Thread {

private Contador contador;

public HiloB (String n, Contador c) {

setName(n);

contador=c;

}

public void run () {

//Bucle para decrementar el contador en 300, sincronizado con el otro hilo

synchronized (contador) {

for (int j=0; j < 300; j++) {

contador.decrementa();

try {

Thread.sleep(100);

} catch (InterruptedException e) {}

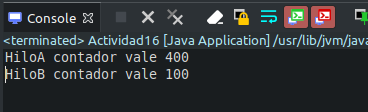
}

System.out.println(getName() + " contador vale " + contador.getValor());

}

}

}

****

**Resolución de problemas**

**REQUISITOS**

1. Entregar las soluciones de las actividades en un único archivo comprimido donde tendremos:
   1. Un archivo de extensión .odt o .docx que contendrá
      1. Una copia del código del programa
      2. Una captura de pantalla del resultado de la ejecución del programa
   2. Los archivos \*.java empleados
2. El único archivo con la soluciones contendrá el nombre original de la actividad, seguido de un guión bajo y del primer apellido del alumno.

**CRITERIOS DE CORRECCIÓN**

1. Cumplimiento de los requisitos (15%).
2. Hacer lo que se indica en el enunciado (55%).
3. Claridad del código Java (15%).
4. Documentación del código Java (15%).
5. Explicación detallada de cómo se han resuelto problemas que hayan podido surgir en la realización de la actividad (hasta un 15% adicional sobre la nota obtenida). Dicha explicación se incluirá al principio de los archivos de código fuente Java mediante líneas de comentarios. La primera línea de comentarios sólo incluirá lo siguiente: // Resolución de problemas.
6. Variables no usadas (-15%).
7. Librerías no usadas (-15%).